

Realization of autonomous image-based spacecraft pointing systems : planetary flyby example

Cheng-Chih Chu, David Q. Zhu, Suraphol Udomkesmalee, and Marc I. Pomerantz

*Jet Propulsion Laboratory
California Institute of Technology
MS 198-326
Pasadena, CA 91109*

ABSTRACT

With a new NASA direction to provide space missions that are low-cost, versatile, and yield maximum science return, future planetary exploration will be accomplished by small spacecraft with a minimum number of components to minimize development and operational costs. This **has** generated a great deal of interest in the development of autonomous spacecraft technology for these future missions. One of the key research area is the development of an autonomous **image-based** spacecraft pointing systems where the **target** position must be measured and estimated **on-board**. By analyzing image data on-board and using it to update the target estimate, the efficiency of the system will **be greatly** improved. In cases of a brief planetary encounter and other asteroid/comet investigation missions, on-board closed loop processing will be essential for maximizing the science return.

This paper summarizes our efforts towards the realization of an intelligent autonomous tracking and pointing system for space applications. A powerful 3D graphic software testbed **has also** been developed to simulate *a* likely scenario of autonomous tracking and pointing operations during a planetary flyby mission.

1. INTRODUCTION

An essential ingredient in the array of new GN&C technologies supporting mission operations autonomy is the ability to identify, classify, and select planetary and celestial features [1]. The interactive functions of autonomous optical navigation and target referenced maneuver guidance and instrument pointing for science remote sensing/imaging will all depend on the development of this capability in both hardware and software. This approach is new to spacecraft target acquisition and image sequence control. Typically, detailed operations such as mosaicking, **slewing**, and new target acquisition, limb tracking, **raster** scanning, and ground track image motion compensation are up-linked to the spacecraft after extensive ground sequence planning and design.

Uncertainties in ephemerides, communications and telemetry analysis time delay, and the inability to respond to unpredictable target opportunities (active sulfur **volcanos** on Io are the archetypical example of this encountered by the Voyager spacecraft) are some of the issues limiting mission return that we face today. Moreover, for missions in which the fly-by time is on the order of the round trip light time, on-board closed loop target tracking is essential to **assuring** successful science return.

A new paradigm for the development autonomous feature-based **spacecraft** pointing systems **was** proposed and described in great details in [1]. At JPL, development is underway to realize such a tracking and pointing system. The

proposed system will be capable of autonomously selecting and classifying planetary and celestial features and updating attitude and guidance references such as body centroids, limb edges, terminators, craters, and other landmarks. The research has been focusing on tracking system architecture, data processing method, development of vision-based algorithms, and the creation of a high-level pointing command language. Progress in this area during the past few years includes an establishment of processing requirements for autonomous image-based pointing systems, and solutions for acquiring/tracking spherical bodies in a fast flyby scenario.

A typical target acquisition/tracking sequence for a planetary flyby mission based on images obtained from a camera field of view consists of the following elements.

1. Distant Encounter:

Single and/or multiple celestial bodies (e.g. the planet and its moon(s)) acquisition and tracking. Here the center of mass (CM) of the planet can be used as pointing reference and target-relative position measurement for spacecraft attitude control and navigation,

2. Near Encounter:

Limb feature extraction for estimating the object size (radius) and spacecraft position. Differentiation of limb from terminator can be done without an a priori Sun position knowledge by examining the edge strength of the object image.

3. Close Encounter:

Autonomous mosaicking of the object activated by having the estimated radius exceeding a threshold computed for a desired $N \times M$ mosaic of the entire planet. Continuous center of mass updates based on visible limb feature also enhance pointing performance and ensure optimal area coverage. Feature extraction and tracking may also be required to obtain necessary target-relative positional information.

In this paper, our efforts in the development of robust, automated, and efficient image processing algorithms for autonomous tracking are presented. All the algorithms developed have been tested extensively based on Voyager image data. Some of test results are shown in this paper. To demonstrate our approach towards the realization of an autonomous image-based tracking and pointing system, a 3D graphic software package is developed to assist the understanding of the likely scenario of autonomous tracking and pointing operations during a planetary flyby mission. The tracking software was successfully demonstrated on a simulated Pluto flyby example.

The rest of this paper is organized as follows. Section 2 discusses relevant image processing techniques needed for autonomous tracking where noise suppression, limb points detection, and least-squares curve fitting for circle and quadratic curves are discussed. In Section 3, some preliminary results for feature detection and tracking are presented, including Hough transform, correlation tracking, and tracking by multiple contour matching.

In Section 4, a representative Pluto Fast Flyby mission scenario is chosen to illustrate the autonomous pointing/tracking technology. In particular, a visualization software was developed on an SGI graphics workstation. The integration of tracking and visualization software provides a powerful graphical demonstration on the realization of an autonomous image-based point/tracking technology for space systems. Finally, some of future works are described.

2. IMAGE PROCESSING TECHNIQUES FOR SPACE APPLICATIONS

The maturity in analytical work as well as the advances in computational algorithms in image processing and computer vision provides us a unique opportunity to apply these concepts to automating the process of obtaining science images during space exploration missions, especially when one considers the radiometric/geometric simplicity of space environments (e.g., dark background, point sources for stars, circular planet bodies, sparse space targets, etc.) in comparisons to earth environments.

To support the development of an intelligent autonomous image-based tracking and pointing system, a great deal of our efforts has been focused on selecting and testing appropriate algorithms for our applications (i.e., space images). Our major emphasis on the selection of algorithms has been focused on automated processing, robustness, and efficiency. For each area, literature survey is usually conducted first to select the candidate algorithms. To aid selection process further, these candidate algorithms are then implemented and tested in a dedicated, X-window based, image processing software system developed by the Autonomous Feature And Star Tracking (AFAST) project at JPL. The testing is conducted based on the criteria stated above using the vast volume of raw data from Voyager's image library. In many cases, modifications to the original algorithms are carried out to improve their performance. In this following, several important areas in image processing relevant to our applications will be discussed and some test results will also be presented.

2.1 Noise suppression

In any image processing application, pre-processing of raw image data is normally required due to the presence of noises or backgrounds. For the "salt-and-pepper" type of noises such as those caused by high energy particles, median filtering is a highly effective method for noise suppression. Other type of low pass filters such as neighborhood averaging would smear the noise and blur the image and consequently prevent accurate extraction of edges [2]. However, median filtering could be computationally intensive. A straightforward implementation of the median filter on an image of size $m \times n$ with filter kernel size $k \times k$ requires $(m - k) \cdot (n - k)$ median search on a $k \times k$ array.

In [3], an efficient method was introduced to eliminate redundancy. A floating histogram is maintained to take advantage of the fact that pixel intensity is an one-byte integer. The histogram of the neighborhood of the very first pixel is determined which is then updated for the next pixel on the same row by removing the leftmost old column and inserting the new column. When index moves to the next column, the histogram is updated by removing and inserting appropriate rows. The merit of this approach is that the computation only increases linearly with the kernel size. (It is possible to optimize a median filter for a particular kernel size.) Similar scheme can be used to implement neighborhood average filter (convolution with a constant kernel) and max-min type of filters (used in edge detection). Details of this fast algorithm can be found in [3].

Figure 2 shows the result of a 5x5 median filtering on a noisy image of Miranda (moon of Uranus) as shown in Figure 1. On a SPARCstation 10/30, it takes 4 seconds approximately to perform a 5x5 median filtering on a 800x800 greyscale image.

2.2 Boundary points extraction (limb detection)

For celestial images, typically, there is a clear separation of the background (the universe) and the object (planets). This corresponds to two well discernible peaks on the intensity histogram of the image and the valley (local minimum) is an appropriate threshold value for segmentation.

For smooth data, the local minimum can be located by differentiating the data (that is numerical differentiation) and searching for zero crossings. However, too many spurious zero crossings will be generated due to noisy data. In [4], a convolution kernel, combining differentiation and smoothing, was introduced to minimize this problem. The histogram is convolved with the convolution kernel (through FFT) to generate a peak detection signal from which the valleys can be found. The algorithm can find multiple valleys by tuning the kernel parameter which adjust the extent of smoothing. If the kernel parameter is smaller, the data is less smoothed and the possibility of detecting more valleys increases. In addition, the detected peaks have to be evaluated against some criteria (such as the width of peaks) to eliminate errors generated by noise.

One characteristic of celestial images is that the background is dominant and the object is small. In this case, the peak value (in the histogram) corresponding to the object will be small and the valley is not deep enough to be detected accurately. The utilization of Laplacian operator has been suggested to increase the symmetry of the histogram [5]. Only pixels which have significant Laplacian (e.g. the top 5%) response are histogrammed. Although Laplacian filter

is known to be sensitive to noises, this problem can be minimized by using the nonlinear Laplace operator proposed in [6].

Another approach to segmentation is modeling each peak in the histogram as a Gaussian and fitting the histogram to multiple-mode Gaussian. This approach requires non-linear optimization which is sensitive to initial values and may not converge to the global extremum.

Once the threshold value is found, it can be used to acquire boundary points by scanning for pixels which has intensity below threshold, and its neighbor has intensity above threshold. To eliminate noise, a continuity check is also performed to ensure that the pixel is from a step edge. Both horizontal and vertical direction have to be checked.

The boundary points thus acquired are not free from noises and may not belong to one single object. Clustering can then be applied to group the boundary points and eliminate noise. For many applications, the simple nearest neighbor clustering is sufficient. For digital images, the Euclidean metric is not always the best choice. Instead, city-block distance or chess-board distance usually works better [7]. Furthermore, either of these 2 metrics are computationally more efficient and represents the digital connectivity more naturally. With a specified maximum separation distance from the nearest neighbor within a group, the algorithm starts from one point and repeatedly extracts points whose distance to one of the points in the cluster is less than the maximum distance. This process is carried out until all points are checked, and then a new cluster is initiated. This algorithm has average complexity $O(n^2)$, where n is total number of points.

A common difficulty in limb detection on the celestial body is the presence of terminators. In our experience working with the many Voyager images, the terminator occurred on the celestial body (especially the spherical body) can be typically characterized by a weaker edge (i.e. lower gradient) as compared to the true limb. Therefore, to detect the limb points in the presence of terminators, a gradient operator (such as Sobel operator) can be used to estimate the gradient of an image first. The edge points are then grouped by their gradients and stronger edge points will be characterized as limb points. This method is intuitively simple but highly effective for the detection of “critical” boundary points. These extracted limb points will typically be used in estimating important geometric parameters of the object, such as the center-of-the-mass location and radius of a spherical body,

It is also important to note that this approach depends on image data only. Although it is not required to have the sun vector information, this method can be made more robust if the sun measurement is available.

2.3 Least-squares curve fitting: circle and quadratic curves

In developing an intelligent autonomous tracking/pointing system for a spherical body, one key element is the capability to provide continuous update on the center-of-mass location and radius of the targeted object. To fit the extracted limb points into a circle, one can pose an optimization problem that minimizes the sum of square errors between the data and the estimates. This is known as the least-squares minimization problem. Although it is a nonlinear problem, the optimal least-squares solution for a circle fitting problem can actually be solved in closed form [8]. Define

$$E(a, b, r) = \sum_{i=0}^{N-1} [r^2 - (x_i - a)^2 - (y_i - b)^2]^2,$$

where (x_i, y_i) , $0 \leq i \leq N-1$ are the points to be fitted. The necessary condition for minimization of $E(a, b, r)$ is

$$\nabla E(a, b, c) = 0,$$

or

$$N r^2 - N a^2 - N b^2 + 2 m_{10} a + 2 m_{01} b - m_{20} - m_{02} = 0 \quad (1)$$

$$m_{10} r^2 - m_{10} a^2 - m_{10} b^2 + 2 m_{20} a + 2 m_{11} b - m_{30} - m_{12} = 0 \quad (2)$$

$$m_{01} r^2 - m_{01} a^2 - m_{01} b^2 + 2 m_{11} a + 2 m_{02} b - m_{21} - m_{03} = 0 \quad (3)$$

where $m_{kl} = \sum x_i^k y_i^l$. From (1),

$$r^2 = a^2 + b^2 - (2m_{10}a + 2m_{01}b - m_{20} - m_{02})/N \quad (-1)$$

By substituting Eq. (4) into Eqs. (2) and (3), the following two linear equations of a and b are obtained

$$(2m_{20} - 2m_{10}^2/N)a + (2m_{11} - 2m_{01}m_{10}/N)b = m_{30} + m_{12} - m_{20}m_{10}/N - m_{02}m_{10}/N, \quad (5)$$

$$(2m_{11} - 2m_{10}m_{01}/N)a + (2m_{02} - 2m_{01}^2/N)b = m_{21} + m_{03} - m_{20}m_{01}/N - m_{02}m_{01}/N. \quad (6)$$

The center (a, b) and the radius r are completely determined.

Similarly, the least-square fit of general quadratic curve can be solved in closed form as well [9]. In general, the quadratic curve is described by

$$Q(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0.$$

But direct minimization of the square error $E(a, b, c, d, e, f) = \sum_{i=1}^{N-1} Q^2(x_i, y_i)$ leads to all coefficients being zero. One way to avoid this is to force one of the coefficients to be nonzero, e.g. let $Q(x, y) = ax^2 + bxy + cy^2 + dx + ey + 1$. The minimization condition $\nabla E(a, b, c, d, e) = 0$ leads to

$$\begin{bmatrix} m_{40} & m_{31} & m_{22} & m_{30} & m_{21} \\ m_{31} & m_{22} & m_{13} & m_{21} & m_{12} \\ m_{22} & m_{13} & m_{04} & m_{12} & m_{03} \\ m_{30} & m_{21} & m_{12} & m_{20} & m_{11} \\ m_{21} & m_{12} & m_{03} & m_{11} & m_{02} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} -m_{20} \\ -m_{11} \\ -m_{02} \\ -m_{10} \\ -m_{01} \end{bmatrix}$$

A better approach is to define a scattering matrix $S = \sum_{i=1}^{N-1} v_i v_i^T$, where $v_i = [x; x_k y_k y_k^2 x_k y_k 1]^T$. S is a real, semi-positive definite matrix. Let $A = [a \ b \ c \ d \ e \ f]$, the matrix A^*SA is minimized by choosing A to be the unit eigenvector associated with the smallest eigenvalues.

A special case of quadratic curves is an ellipse. The condition for a conic section to be an ellipse is $b^2 - 4ac < 0$. The orientation of the ellipse is given by $\alpha = \frac{1}{2} \tan^{-1}(\frac{b}{a-c})$. Applying coordinate transformation, the regular equation of an ellipse is obtained:

$$\frac{(x + \frac{D}{2B})^2}{\frac{1}{B}(\frac{D^2}{4B} + \frac{E^2}{4C} - F)} + \frac{(y + \frac{E}{2C})^2}{\frac{1}{C}(\frac{D^2}{4B} + \frac{E^2}{4C} - F)} = 1$$

where

$$\begin{aligned} B &= a \cos^2 \alpha + b \sin \alpha \cos \alpha + c \sin^2 \alpha \\ C &= a \sin^2 \alpha - b \sin \alpha \cos \alpha + c \cos^2 \alpha \\ D &= d \cos \alpha + e \sin \alpha \\ E &= -d \sin \alpha + e \cos \alpha \\ F &= f. \end{aligned}$$

It is noted that the best estimates of various parameters in either circle or quadratic curve-fit can be found with very little computation effort. Figure 3 shows the result of fitting a circle to the limb of the object in Figure 2. The corresponding result for a best elliptical fit is shown in Figure 4.

3. FEATURE DETECTION AND TRACKING

Feature detection and tracking is one of the most interesting and difficult problems in developing an intelligent autonomous imaged-based tracking and pointing system. During a near encounter, it is likely that only a close-up, frame-filling view of the planet will be available. In this case, a robust on-board feature-based tracking system will be required to extract necessary target-relative positional information by matching with a model of known surface features or terrains (e.g., the red spot on Jupiter's surface). In addition, such on-board capabilities will enable the spacecraft not only to respond to unpredictable target opportunities (such as active sulfur volcanos on Io), but also to search for interesting science targets if it is instructed to do so. Therefore, the science return can be greatly enhanced. In this section, some of our work along this research direction will be discussed and preliminary results will also be presented.

3.1 Circular feature detection using Hough transform

The Hough transform is known for its robustness, but also for its computational complexity and large memory usage in contrast to other approaches (such as fuzzy clustering in [10]). The principle of the Hough method is to transform image points to the parameter space and search for clusters in the parameter space. Unfortunately, the complexity grows exponentially with the number of parameters. Through careful implementation, the Hough transform is feasible for detecting lines and circles. An excellent summary of all the techniques developed in circle detection by the Hough transform can be found in [11].

Since the rediscovery of the Hough transform [12], a major advance was using gradient to reduce the complexity. The gradient of an image point indicates the relevant range of parameters to accumulate, so it can also increase the accuracy. The high order and separable Sobel operator [13] can estimate the gradient accurately and efficiently. In the circle detection, parameters for the center are first accumulated and the peaks (the center candidates) are detected in this 2-dimensional parameter space (The high order Laplacian filter can be used to enhance the peaks). Then radii for each center are histogrammed and the peaks (the radii candidates) are detected in the 1-dimensional array.

One important class of features commonly found on the celestial bodies is the crater which appears to be circular along the edge for a view directly above. Hough transform could be a useful technique capable of detecting multiple craters simultaneously. Figure 5 shows a distant image of Mimas (moon of Saturn) where a large crater appears on its surface. The Hough transform for circle detection was applied to this image, and the crater and the limb of Mimas were detected successfully as shown in Figure 6.

Hough transform has also been extended to detect ellipses, and even arbitrary shapes [14] which can potentially be applied to detect other interesting celestial features such as ridges, rivers, and Jupiter's red spot.

3.2 Feature tracking by correlation

Correlation is a straightforward method for template matching. But it is sensitive to size, orientation and lighting. The lighting problem can be reduced by subtracting the average; this leads to the following formulation

$$\gamma(x, y) = \frac{\sum_j \sum_i [I(i, j) - \bar{I}(i, j)][T(i - x, j - y) - \bar{T}]}{\sqrt{\sum_j \sum_i [I(i, j) - \bar{I}(i, j)]^2 \sum_j \sum_i [T(i - x, j - y) - \bar{T}]^2}}$$

where $I(i, j)$ is the image pixel values and $T(i, j)$ is the template pixel values. The largest coefficient $\gamma(x, y)$ corresponds to the best match to the template in the image. Correlation may be more efficiently carried out in the frequency domain via FFT, especially for large template [2].

The correlation technique is known to be sensitive to noises and scale/perspective variation, and, computationally intensive [1,13]. Nevertheless, it could still be effective if it is used properly. As an example, correlation tracking was applied successfully to a sequence of 4 Jupiter's images selected from Voyager's image library where the tracked feature

is a closed contour appeared on the Jupiter. The results are shown in Figure 7 through 10. It is our conjecture that the correlation would work well for local feature tracking if the time between image frames is relative short, and hence, the typical problems associated with the correlation method such as scale and perspective changes, can be minimized. Further study in this area is required.

3.3 Feature tracking by multiple contour matching

In [16], an alternative method for feature tracking **was** proposed where tracking of planetary surface terrains is done by recognizing the pattern in "feature constellations". The **fundamental** concept of this method is similar to those employed in recognizing the pattern among star constellations in star identification and tracking problems.

To implement such an approach, closed contours need to be detected first. The procedure for contour detection is well-known [2]. For example, one can first remove the undesired noise and background from the raw image using median filtering. The gradient-based edge detection technique such as Sobel filtering can then be used to detect candidate boundary points which can be further reduced by edge thinning to simplify the detection of closed-contours. The scheme of surrounding-edge contour detection described in [1] can then be applied to detect all possible closed contours within the same image. These contours are then cataloged to provide a reference for the subsequent image frame. **When** a new image frame is acquired, closed-contours are detected and listed. Finally, pattern matching using relative distance and individual characteristics (e.g., edge strength, enclosed area, etc.) can then be performed to identify the best match.

To demonstrate the **feasibility** of this approach, two different close-up images of Jupiter surface are selected from Voyager's library as shown in Figure 11 and 12. The detected closed-contours in each image are highlighted respectively. As a result of this proposed approach, a pattern with multiple features (contours) **was** matched successfully. Although it is still very preliminary, the test results are very encouraging. It is our belief that this novel approach will have great potentials in the area of feature tracking.

4. AUTONOMOUS TRACKING AND POINTING DEMONSTRATION

To demonstrate our approach towards the realization of intelligent autonomous tracking and pointing systems, a 3D graphics software testbed **has** been developed by the AFAST project at JPL. This testbed not only allows us to validate our algorithmic design using the dynamic scenes generated by the graphics functions, but also provides a great visualization tool to simulate the likely scenario of autonomous tracking and pointing operations **during** a planetary flyby mission. Specifically, a simulated **Pluto flyby** scenario [17] **has** been implemented to illustrate the capabilities of the proposed intelligent autonomous tracking system.

The demonstration is implemented in a distributed simulation **environment** consisting of two processes, the visualization software, and the tracking software. In the following, we will describe each of the two software modules and the interaction between **these** two modules.

4.1 Visualization system overview

The AFAST visualization software runs **on** a Silicon Graphics high speed graphics workstation (IRIS Crimson, VGXT) and utilizes the built-in graphics hardware of the workstation to perform the realistic texture mapping, lighting, shading and general graphics transformations needed to generate the realism required for this demonstration.

The software is a completely event-driven system that was **designed** using an object-oriented paradigm and was implemented in C++ and SGI/GLTM. The system was designed to be used as a general visualization tool and is not specific for the AFAST project. To achieve this generality, the software is completely data-driven. Ephemeris data, view port size and location, spacecraft and planet models, and spacecraft trajectory information are described in a configuration **file** that is parsed by the software at run-time. In addition, the user can easily reconfigure the viewing

location, line-of-sight vector, and field-of-view (FOV). The corresponding scene can be placed at any desired location with the chosen size on the graphic console. Here the viewer can be used in a very general sense which could be the user sitting in front of the display and/or any sensing instrument (such as imaging camera and star tracker) located on the spacecraft. Currently, the only way to change viewing information is through the modification of the configuration file. However, a Graphical User Interface (GUI) will be implemented in the future.

4.2 Tracking software

The tracking software module was designed to perform the required image data process autonomously, that is, no user-input parameters is allowed. For example, the algorithms must find the threshold value automatically before boundary points are extracted from the image. This has been a major driver in the process of selection, modification, and implementation of processing algorithms.

There are several major elements in the current tracking software which can be easily expanded.

1. Searching:

Initially, the target must be acquired through search due to the uncertainty in the knowledge of its location. The searching operation essentially is the mosaicking of sky in a restricted area governed by the target's positional uncertainty. The mosaic size is determined then by the sensor's field-of-view (fov), the size of search area, and the overlap between search windows. Assume that the uncertainty u is the same in both x and y direction of the sensor coordinate frame, the number of search window is determined by

$$n = \frac{\frac{u}{fov} - \alpha}{1 - \alpha}$$

where α is the overlap as a fraction of fov . The clustering technique discussed in Section 2.2 can then be used to detect the target(s). The search terminates when the object is detected.

2. Tracking:

If more than one significant target is detected, each object is fitted to a circle and their geometric center is tracked. For example, if two bodies are detected, the point (x_t, y_t) is tracked where $x_t = x_1 + \frac{r_2(x_2 - x_1)}{r_1 + r_2}$, $y_t = y_1 + \frac{r_2(y_2 - y_1)}{r_1 + r_2}$, and $\{(x_i, y_i), r_i\}$ ($i = 1, 2$) are the estimated circle center and its corresponding radius, respectively. As the spacecraft approaches the targets, all the objects will no longer be observed within the same FOV. In choosing the tracking point this way, the larger body will always be presented in the sensor's FOV. Of course, the tracking point can be varied depending on the mission design.

Once the smaller body is moved out of the field of view. The center of the larger body is tracked and the radius is monitored. When the radius becomes larger than certain pre-specified value (60% of the FOV), it is necessary to track the limb in order to continuously obtain an accurate estimation of the center (x_c, y_c) and the radius " r " of the target. Since the terminator is a very typical phenomenon in the planetary mission, one should be careful in choosing the proper limb for tracking. For example, if the lower left limb is tracked,

$$x_t = x_c - \frac{r}{2}, \quad y_t = y_c + \frac{r}{2}$$

can be used as the tracking point. A better approach would attempt to capture the longest limb possible. It requires that the limb passes through the two opposite vertices of the image window. Then the tracking point for this case is determined by

$$x_t = x_c - \frac{h}{\sqrt{2}}, \quad y_t = y_c + \frac{h}{\sqrt{2}}.$$

where $h = \sqrt{r^2 - \frac{w^2}{2}}$, w is FOV in pixels. Another choice is to track the intersection of the circular limb and the diagonal of the image window can be tracked. The tracking point (x_t, y_t) for this case is determined by the

solution of

$$(x_t - x_c)^2 + (y_t - y_c)^2 = r^2, \quad x_t - y_t = 0$$

or

$$(x_t - x_c)^2 + (y_t - y_c)^2 = r^2, \quad x_t + y_t = w$$

3. Mosaicking:

When the radius of the target reaches certain pre-specified threshold value of the sensor's FOV, a certain size of mosaic is carried out autonomously. The tracking center is determined by

$$x_t = x_c + m_x(r + \delta r)(1 - f_{overlap}), \quad y_t = y_c + m_y(r + \delta r)(1 - f_{overlap})$$

where m_x and m_y are the location of the mosaic window with respect to the center of the planet, δr is the radius change from the previous estimation and $f_{overlap}$ is the overlap between mosaic window as a fraction of FOV. In case of 3x3 mosaic, 80% of the FOV is used as the threshold value to trigger the mosaic sequence. Ideally, this threshold value can be optimized for a given mosaic size. For a 3x3 mosaic, only the frame in the center captures no limb, so no information about the center and radius is available and it simply uses information propagated from the previous estimation. If feature detection and tracking capability is added, a more precise mosaic is possible and it is very desirable if the size of mosaic is large.

By processing the image, the desired tracking pointing (x_t, y_t) (e.g., center-of-mass, or limb) on the image plane is estimated. The offset of the tracking point from the center of the image plane has to be computed in terms of rotation angles. Based on the image, only rotation angle about x and y can be approximated by

$$r_x = \frac{fov_y(y_t - \frac{h}{2})}{h}, \quad r_y = \frac{fov_x(x_t - \frac{w}{2})}{w}$$

(the boresight is along the z-axis), where w and h is the image width and height, respectively. Furthermore, an additional correction due to compensate spacecraft's motion is also estimated and combined with the correction from the image.

4.3 Integration of visualization and tracking software

To perform the various tracking functions described previously, the visualization software must communicate with the tracking software to properly orient the spacecraft during the simulation of the entire flyby. During the execution of the simulation, the tracking software continually prompts the visualization software for current views through the simulated sensor (such as the imaging camera). Whenever a new view is requested, the visualization software saves the current image as defined by the sensor's FOV to a buffer and subsequently sends the contents of that buffer, via custom TCP/IP messaging software, to the tracking software. The tracking software will then process the image using the algorithms in the tracking software to compute the required correction in spacecraft orientation in order to achieve the desired pointing direction for the sensor. This information is then sent back to the visualization software. The requested correction in spacecraft orientation is then executed, and a new scene is generated.

Since the operation of the tracking and visualization software is tightly coupled, both processes must be synchronized to assure that the information provided by the tracking software can orient the spacecraft correctly. This synchronization is achieved via timing messages sent from the tracking software to the visualization software. In this way, the tracking software controls the movement of simulation time, and could conceivably run the simulation backwards if that were desired.

Note that only the visualization software needs to be hosted on the graphics workstation, the tracking software can be running either on the same CPU or on a different CPU. For the latter case, the additional CPU could be either the second processor on the same workstation or another workstation. Hence, the simulation process can be speeded significantly.

4.4 Demonstration – Pluto flyby example

This demonstration shows our capabilities in multiple-body searching/detection/tracking, center-of-mass tracking, limb tracking, and autonomous mosaicking in a simulated Pluto flyby environment. One key parameter in this simulation is the FOV of the imaging camera which is specified in the configuration file. With a given set of trajectory data, multiple bodies may not be observed in the same image if the FOV is small. On the other hand, mosaicking may not be necessary if the FOV is large. For this demonstration, a 3 degrees by 3 degrees FOV is used in order to show all the capabilities mentioned above.

Initially, the spacecraft is located approximately 660,000 Km away from Pluto (11 hours before the closest encounter to Pluto) and the velocity vector of the spacecraft is assumed known. At this distance and the closeness of Pluto and its moon - Charon, both bodies can be observed within the FOV. However, due to the uncertainty in the positional knowledge of the Pluto-Charon system, the spacecraft will turn and search a specific area governed by the uncertainty in order to detect the desired targets. Once they are detected, as the spacecraft approaching Pluto, tracking of the two bodies will be carried out continuously until both Pluto and Charon will no longer fit into the same FOV. Under this circumstance, tracking of the center-of-mass of the larger body (Pluto) is exercised.

When the estimated image diameter of the planet is greater than camera FOV, limb tracking is carried out. This will allow the on-board image processing algorithms to estimate and update the center of mass location and the radius continuously.

Finally, when the estimated image diameter of the planet exceeds a preset threshold, a 3x3 mosaic is performed automatically (Note that the size of mosaic can be changed based on the mission design). After mosaicking, limb tracking will continue. The simulation ends approximately one half hour after closest encounter.

All the key steps of this simulation were summarized in Figure 13, and a video demonstration of the entire simulation will also be presented.

5. CONCLUSION AND FUTURE WORK

The feature-based pointing technology proposed in [1] will be one of the major contributors to achieving the goals of the new NASA directive: to provide space missions that are low-cost, innovative, versatile, and yield the maximum science return. With the advances in image processing and computer vision, the realization of robust, autonomous image-based tracking and pointing technology on spacecraft systems can be considered as a real possibility. In this paper, our work in achieving such a goal is reported. Our approach to the realization of an intelligent autonomous image-based pointing and tracking system was also demonstrated successfully. Specifically, a 3D graphics package was developed and a planetary flyby example was implemented to show our capabilities in searching/detecting, tracking, and autonomous mosaicking of spherical bodies.

For our future plan in the near term, a natural step is to extend the work in this paper to the acquisition, tracking, and mosaicking of the irregular-shaped bodies such as asteroids and comets. Another related area in autonomous tracking and pointing operations is surface feature detection and tracking which is crucial in a close-up operation when no limb information is available. Although some preliminary results were reported in this paper, more research efforts need to be carried out in this area.

ACKNOWLEDGMENTS

The authors would like to express their thanks to Mr. Mark Garcia and Mr. James Duke for their assistance in providing data for the graphics demonstration. This research was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. S. Udomkesmalee, G.E. Sevaston, and R. II. Stanton, "Toward an autonomous feature-based pointing system for planetary missions", *SPIE* Vol. 1949, Space Guidance, Control, and Tracking, 1993.
2. R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
3. T.S. Huang, G.T. Yang, and G.Y. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. on ASSP*, Vol. ASSP-27, No. 1, February, 1979, pp. 13-18.
4. M.I. Sezan, "A peak detection algorithm and its application to histogram-based image data reduction," *CVGIP*, 49, 1990, pp. 36-51.
5. J.S. Weszka, R.N. Nagel, and A. Rosenfeld, "A threshold selection technique," *IEEE Trans. On Computer*, 23, 1974, pp. 1322-1326.
6. L.J. van Vliet, I.T. Young, and G.L. Beckers, "A nonlinear Laplace operator as edge detector in noisy images," *CVGIP*, 45, 1989, 167-195.
7. A. Rosenfeld and A.Y. Wu, "Digital geometry on graphs," *Contemporary Mathematics*, Volume 119, 1991, pp. 129-136.
8. S.M. Thomas and Y.T. Chan, "A simple approach for the estimation of circular arc center and its radius," *Computer Graphics Image Processing*, 45, 1989, pp. 362-370.
9. A. Albano, "Representation of digitized contours in terms of conic arcs and straight line" segments," *Computer Graphics Image Processing*, 3, 1973, pp. 23-33.
10. R.N. Dave, "Generalized fuzzy c-shells clustering and detection of circular and elliptical boundaries," *Pattern Recognition*, 25, 1992, 713-721.
11. P. Kierkegaard, "A method for detection of circular arcs based on the Hough transform," *Machine Vision and Applications*, No. 5, 1992, pp. 249-263.
12. R.O. Duda and P.E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *Communication ACM*, Vol. 15, No. 1, 1972, pp. 11-15.
13. P.E. Danielson and O. Seger, "Generalized and separable Sobel operators," H. Freeman (cd), *Machine Vision-Acquiring And Interpreting The 3D Scene*, Academic Press, New York, 1989, pp. 347-379.
14. D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, Vol. 13, No. 2, 1981, pp. 111-122.
15. E.L. Hall, *Computer Image Processing and Recognition*, Academic Press, 1979.
16. C.C. Liebe, "A new strategy for tracking planetary terrains," *SPIE*, Paper No.. 2221-68, (this proceeding), April 5-8, 1994.
17. R.L. Staehle, J.B. Carraway, C.G. Salvo, R.J. Terrile, S.S. Weinstein, and F. Hansen, "Exploration of Pluto: search for applicable satellite technology," *Sixth Annual AIAA/Utah State University Conference on Small Satellites*, Logan, Utah, September, 1992.

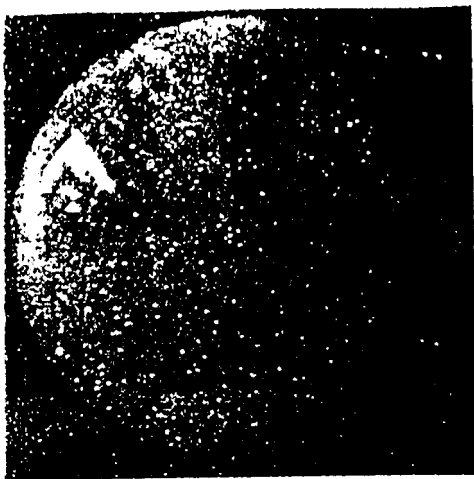


Figure 1: Noisy image of Miranda



Figure 2: 5x5 median filtering

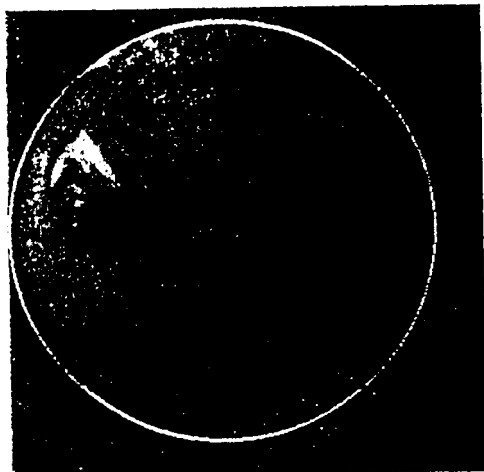


Figure 3: Circle-fit of Miranda

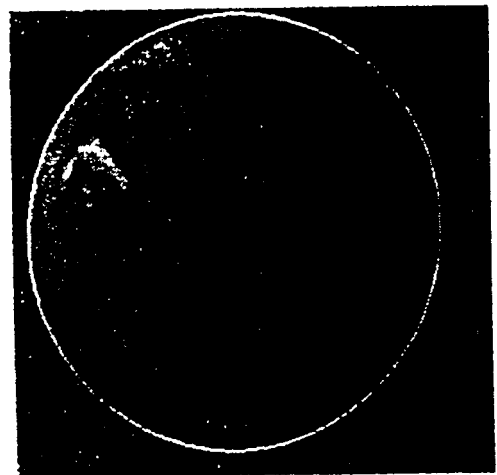


Figure 4: Ellipse-fit of Miranda

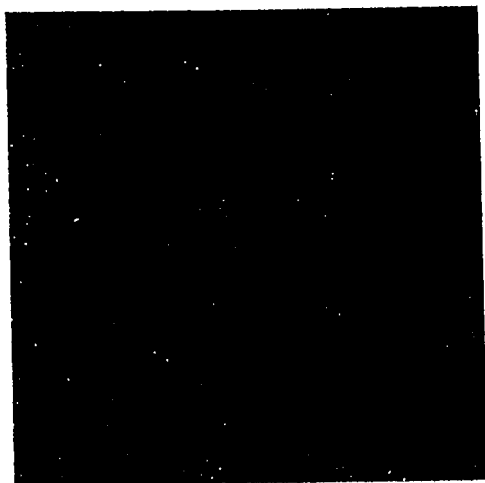


Figure 5: Mimas

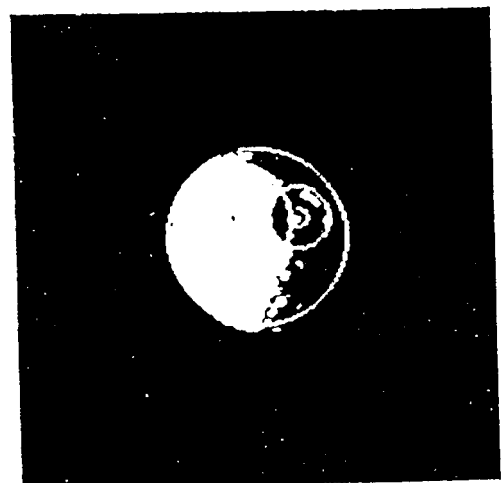


Figure 6: Circular feature detection on Mimas

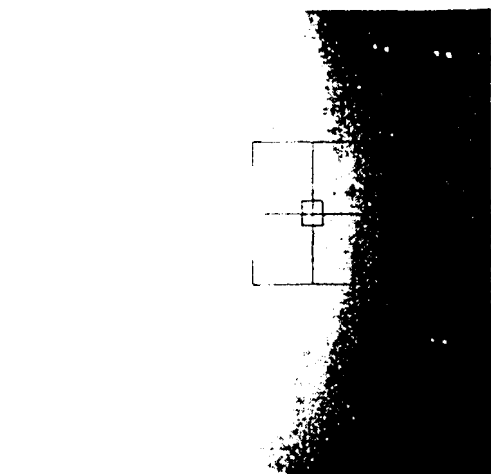


Figure 7: Correlation feature tracking - frame #1

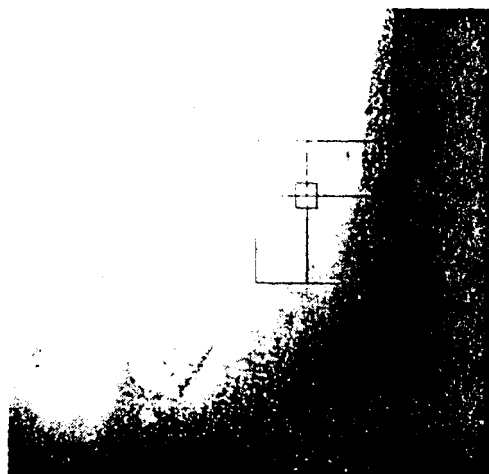


Figure 8: Correlation feature tracking - frame #2

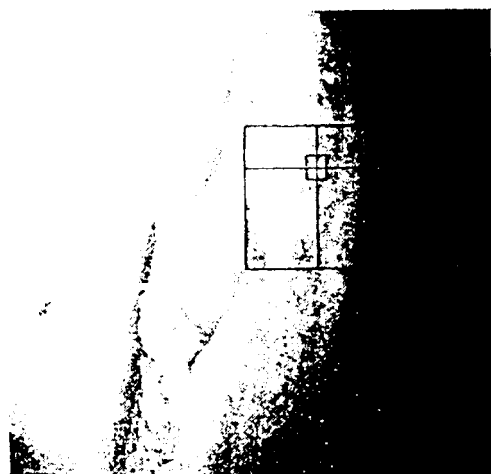


Figure 9: Correlation feature tracking - frame #3

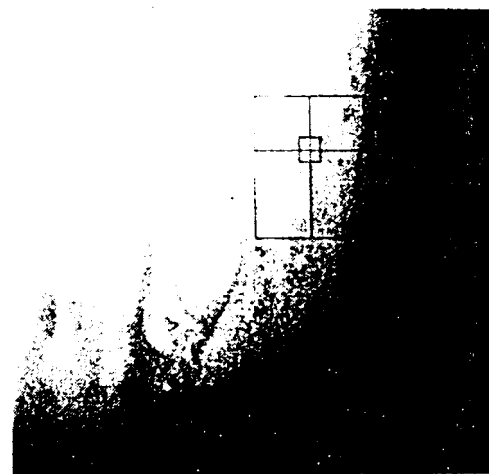


Figure 10: Correlation feature tracking - frame #4

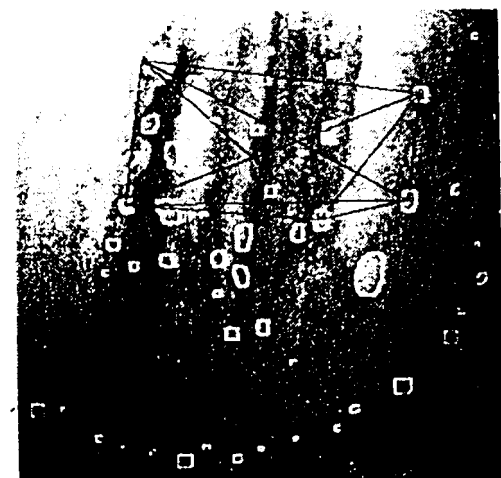


Figure 11: Multiple contour matching - frame #1

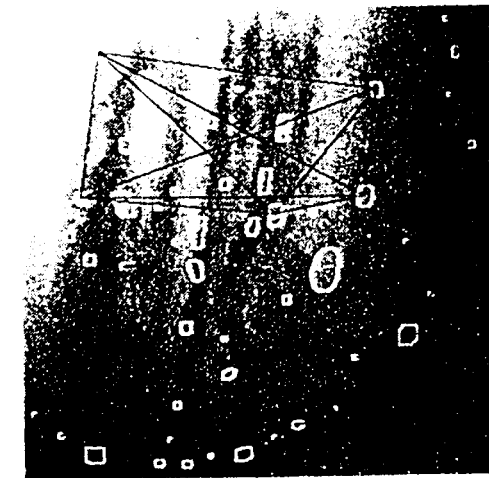


Figure 12: Multiple contour matching - frame #2

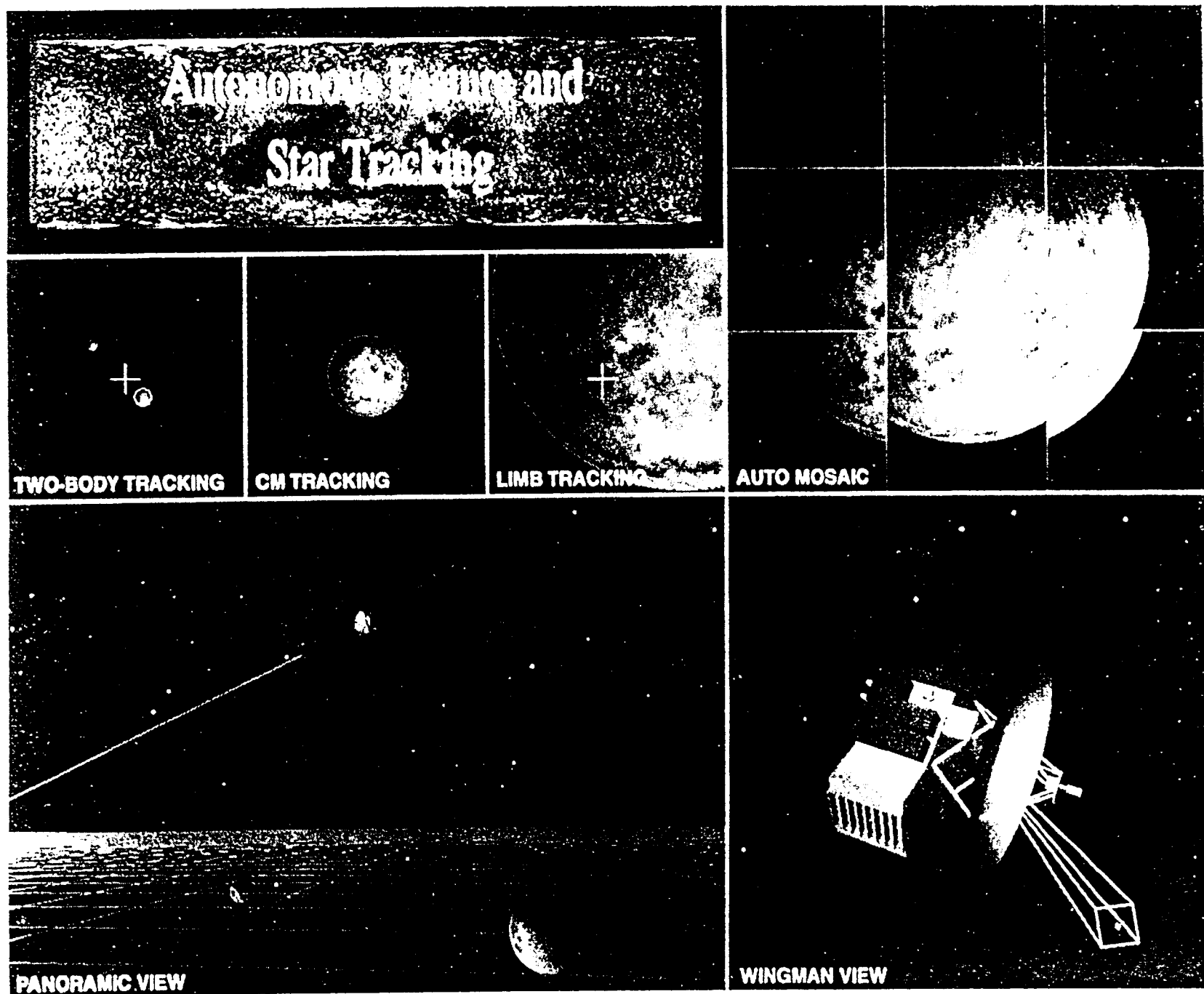


Figure 13: Autonomous tracking and pointing demonstration: Pluto flyby example